'18 MARCH

01 Wk 09 • 060 Day THURSDAY

March 2018

Wk	M	T	W	T	F	S	S
09				1	2	3	4
10	5	6	7	8	9	10	11
11	12	13	14	15	16	17	18
12	19	20	21	22	23	24	25
13	26	27	28	29	30	31	

Applied Machine Learning (in python)

(Univ. of Michigan)

Scikit-Learn
SciPy
NumPy
Pandas
matplotlib

Visualization :
cm.get_cmap('gnuplot')
(small no. of features)

→ 3d → Axes3D
from mpl_toolkits.mplot3d

K-Nearest Neighbour.
(for regression & classification)
↳ query point
↳ decision boundary.

→ Euclidean dist (Minkowski with Power Parameter p=2)

accuracy :- .score()

Supervised Learning - (Label)

feature representation
Data instances / samples / examples (x)
Target value (y)

April 2018

Wk	M	T	W	T	F	S	S
13	30				6	7	8
14	2	3	4	5	13	14	15
15	9	10	11	12	20	21	22
16	16	17	18	19	27	28	29
17	23	24	25	26			

MARCH 18

Wk 09 • 061 Day

FRIDAY 02

Classification & Regression.

Discrete target — continuous.

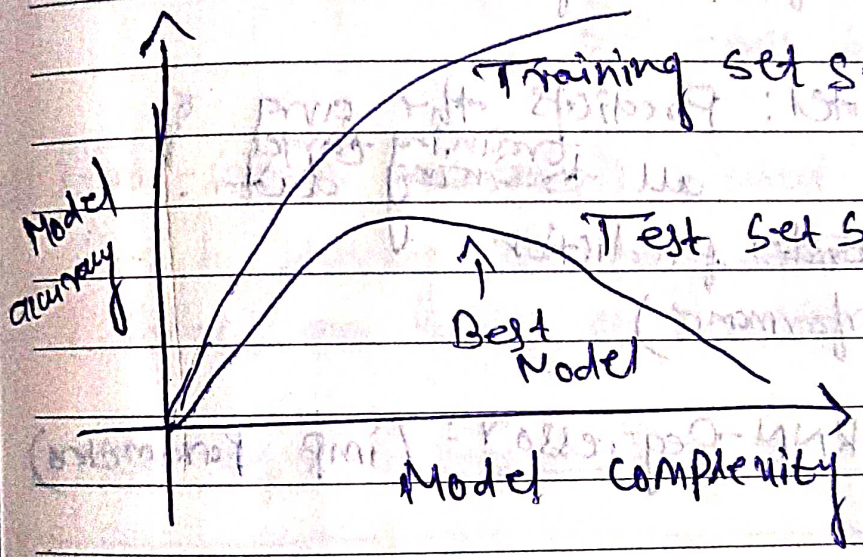
eg: Yes/No

1/0

Multiclass classification.

(~~More discrete values~~)

Multi-Label classification



Overfitting & Underfitting :

for KNN → K = 1 → Overfitting.
K = 5 → good
K = 10 → Underfitting

→ for the last example.

K ↓ Risk of overfitting ↑ for KNN

KNN can be used for both classification & Regression.

R^2 - regression score ∈ [0,1]
(co-efficient of determination)

→ Measures how well the prediction model for regression fits the given data.

0 - const model: Predicts the avg of all training target data.

1 - Perfect predictor.
(best performance)

KNN-Classifier & KNN-Regressor: (imp Parameters)

Model_complexity:

n_neighbours : default = 5.

04 Sunday

Model_fitting:

metric: dist fun^n btw data points.
default = p = 2.

April 2018

Wk	M	T	W	T	F	S	S
13	30						1
14	2	3	4	5	6	7	8
15	9	10	11	12	13	14	15
16	16	17	18	19	20	21	22
17	23	24	25	26	27	28	29

MARCH '18

Wk 10 • 064 Day

MONDAY 05

Least-Square : L.R.

Input instance (feature vector) : $x = (x_0, x_1, \dots x_n)$

Predicted output : $\hat{y} = \hat{w}_0 x_0 + \hat{w}_1 x_1 + \dots \hat{w}_n x_n + \hat{b}$

Parameters to estimate : $\hat{w} = (\hat{w}_0, \dots \hat{w}_n)$: feature weights / model coefficients.

$\hat{b}$: const bias term / intercept.

Least-Square method : is used to estimate $\hat{w}$ & $\hat{b}$ during training.
(There are also other methods)

mean square Error = $(x_i - \bar{x})^2$

In Linear Regression, there is no parameter to control Model complexity.

obj fun^n - optimize
Loss fun^n - minimize

$RSS(w,b) = \sum_{i=1}^{N} (y_i - \hat{y})^2$; $\hat{y} = (w \cdot x_i + b)$

(Sum of Square Diff) (RSS)

(Residual Sum of Squares) (RSS)

sklearn : LinearRegression

w_i = linreg.coef_

b_i = linreg.intercept_

((Underscore denotes a quantity derived from training data,

LR: Ridge, Lasso & Polynomial Regression

$$RSS_{Ridge}(w,b) = \sum_{i=1}^{N} (y_i - (w.x_i + b))^2 + \alpha \sum_{j=1}^{p} w_j^2$$

Regularization

↓

It prevents overfitting by: restricting model, (reducing its complexity)

L2 Regularization: min sum of sq. of w entries.

α↑ regularization ↑ ⇒ more simple model.

$\alpha = 1$ (default)

```
# from sklearn.linear_model import Ridge,
```

feature pre-Processing & Normalization.

MinMax scaling. $[0,1]$ -> value.

$$x_i' = \frac{x_i - x_i^{Min}}{x_i^{Max} - x_i^{min}}$$

from sklearn.preprocessing import MinMaxScaler

transform().

*Data Leakage.

Small no. of training data compared to no. of feature ⇒ Regularization works well.

Lasso Regr—

$$RSS = \text{_____} + \lambda \sum_{j=1}^{p} |w_j|$$

L1 Penalty

Sparse solution: Kind of feature selection.

from _____ import Lasso

'18 MARCH

08 WK 10 • 067 Day
THURSDAY

March 2018

Wk	M	T	W	T	F	S	S
09				1	2	3	4
10	5	6	7	8	9	10	11
11	12	13	14	15	16	17	18
12	19	20	21	22	23	24	25
13	26	27	28	29	30	31	

Polynomial feature Transformation:

$x = (x_0, x_1) \rightarrow x' = (x_0, x_1, x_0^2, x_0 x_1, x_1^2)$

Non-linear basis functions.

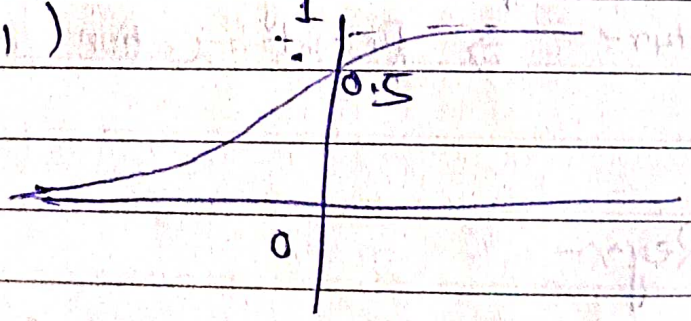
Logistic Regression :-

(Used for Classification)

$$\hat{y} = \text{logistic}\left(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n x_n\right)$$

$$= \frac{1}{1 + \exp\left[-\left(\hat{b} + \hat{w}_1 x_1 \cdots\right)\right]}$$

$y \in (0, 1)$



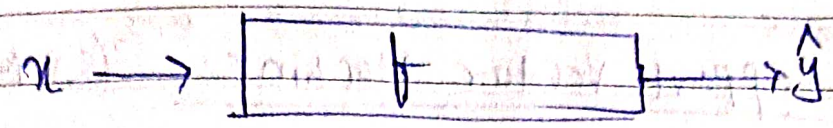
from sklearn.linear_model import LogisticRegression

✓ L2 regularization is 'on' by default.

✓ Parameter C controls amount of regularization. (default 1.0)

↑ C ⇒ Regularization ↓

Linear classifiers : SVM :



$f(x, w, b) = \text{sign}\ (w \cdot x + b)$

dot product.

$(w_1, w_2) \cdot (x_1, x_2)$

$= w_1 x_1 + w_2 x_2$

→ Classifier Margin :

Max width the decision boundary can be increased before hitting a data point.

→ Max margin classifier :

Linear classifier with max margin is a Linear SVM (LSVM)

from sklearn.svm import SVC.

* Larger C : less regularization.

→ fit the training data as well as possible

→ each individual data point is imp to classify correctly.

* Smaller C : More regularization.

→ More tolerant of errors on individual data points.

18 MARCH

10 Wk 10 • 069 Day
SATURDAY

March 2018

Wk	M	T	W	T	F	S	S
09				1	2	3	4
10	5	6	7	8	9	10	11
11	12	13	14	15	16	17	18
12	19	20	21	22	23	24	25
13	26	27	28	29	30	31	

Multiclass classification:

Kernelized Support Vector Machine: (SVM)

(both classification & Regression)

↳ RBF: Radial Bajs function kernel.

↳ Polynomial kernel.

RBF.

$$k(x, x') = \exp\left[-\gamma \| x - x' \|^2\right]$$

↑

gamma (γ) : kernel width, Parameter.

- Kernel type.
- γ value
- 11 Sunday C - Parameter.

→ These 3 affect kernel Complexity.

Cross Validation:

k-fold cross validation.

from sklearn.model_selection import cross_val_score

Stratified k-fold Cross validation.

→ Leave one out cross validation : k=1 data for test set.

→ Validation curve

Decision Trees

(Supervised)

→ If then conditions : series of yes/No questions.

→ Root Node, Leaf Node.

from sklearn.tree import
DecisionTreeClassifier

Controlling the Model Complexity :
Parameters : ↳ max_depth
↳ Max # of leaf nodes = max_leaf_nodes.
↳ Min sample to
consider splitting = min_samples_leaf

'18 MARCH

13 Wk 11 • 072 Day TUESDAY

Re-Read : SVM
Ridge Regression
Lasso Regression

March 2018

Wk	M	T	W	T	F	S	S
09				1	2	3	4
10	5	6	7	8	9	10	11
11	12	13	14	15	16	17	18
12	19	20	21	22	23	24	25
13	26	27	28	29	30	31	

feature importance :
(How imp is a feature to overall prediction accuracy)

f.i = 0 : Not used in prediction.
1 : the feature predicts the target perfectly

Model Evaluation & Selection

Accuracy also has drawbacks.

#Imbalanced class.

Dummy Classifier :
↳ Provide a null metric (eg: null accuracy) baseline.
↳ Dummy Classifier should not be used for real problems.
↳ Accuracy achieved by always picking the most frequent class.
↳ It ignores the input data.

Common settings for the "Strategy" parameter for DummyClassifier in SKlearn :
↳ Most_frequent : Predicts most freq. label.
↳ Stratified : random pred.

↳ Uniform: generates Pred uniformly at random.
↳ Const: Always Pred a Const label provided by user.

AUC (Area Under the Curve):

Confusion matrix:

from sklearn.metrics import Confusion_matrix

True -ve	False +ve
False -ve	True +ve

← Confusion matrix order

False +ve = Type-1 error
False -ve = Type-2 error

Confusion Matrices & Basic Evaluation Metrics:

* Classification Error = $\frac{\#\text{Errors}\ (FP+FN)}{\#\text{total}\ (FP+FN+TP+TN)}$

(1 - Accuracy)

* Recall = $\frac{TP}{TP+FN}$ (frac" of all +ve instns does the classifier correctly identify as +ve)

* Precision : from^ of the predictions are correct? $\frac{TP}{TP+FP}$

* False Positive Rate (FPR) : (or Specificity) $\frac{FP}{TN+FP}$

F1-score : HM of Precision & Recall.

$$F1 = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}$$

F1 is a case of F-score.

$$F\text{-score} = (1+\beta^2) \frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall}$$

#from sklearn.metrics import accuracy_score, Precision_score, recall_score, f1_score, classification_report_score.

Classifier Decision functions:
decision_function().
Predict_Proba().

Precision Recall & ROC curves:

↓

Receiver Operating Characteristic Curves

Multi-Class (~~classification~~ Evaluation):

micro vs Macro Average.

↓

(each class has equal weight)

↓

Each instance has equal wt.

Largest classes have most influence.

↳ If classes have ~~about~~ same no. of instances,
macro ≈ micro

↳ If some classes have much larger (more instances) than others :

wt metric towards largest one - micro

wt " " smallest " - macro

↳ micro << macro, Larger class has poor metric Perf

↳ micro >> macro, Smaller " "

'18 MARCH

17 Wk 11 • 076 Day SATURDAY

March 2018

Wk	M	T	W	T	F	S	S
09				1	2	3	4
10	5	6	7	8	9	10	11
11	12	13	14	15	16	17	18
12	19	20	21	22	23	24	25
13	26	27	28	29	30	31	

Regression Evaluation :

Optimizing Classifiers for different Eval. metrics:

data leakage :-

Dimensionality Red & Manifold Learning:

PCA: Principal Component Analysis

from sklearn.decomposition import PCA

MDS - Multidimensional scaling,

from sklearn.manifold import MDS

TSNE

Clustering :

18 Sunday

Naive Bayes Classifier :

3 types (available in sklearn)

↳ Bernoulli : binary features. (eg : word present or not)

↳ Multinomial : discrete features (eg : word counts)

↳ gaussian : Continuous /real valued features.

↳ # from sklearn.naive_bayes import GaussianNB

supports partial fit.

Random Forest :

⇒ An ensembles of trees.

sklearn.ensemble module :

classificator : RandomForestClassifier

Regression : RandomForestRegressor

n_estimator Parameter.

Bootstrap Samples

max_features

Gradient boosted decision trees :

(weak learners)

(learning rate)

Neural Networks :

Multi Layer Perceptron

(feed forward N.N)

(tanh activation func^n)

#Import MLPRegressor from sklearn.neural_network

Imp parameters :

↳ hidden_layer_sizes

↳ alpha

↳ activation

Data Leakage